UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/589,239	01/03/2007	Ivan Boule	LSN-4786-8	6941

23117 7590 04/06/2011

NIXON & VANDERHYE, PC
901 NORTH GLEBE ROAD, 11TH FLOOR
ARLINGTON, VA 22203

EXAMINER	
SADLER, NATHAN	
ART UNIT	PAPER NUMBER
2189	

MAIL DATE	DELIVERY MODE
04/06/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/589,239	BOULE ET AL.	
	Examiner	Art Unit	
	Nathan Sadler	2189	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 March 2011*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-9,11-24,26-38,40-46 and 48-50* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-9,11-24,26-38,40-46 and 48-50* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 112

1. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

2. Claims 1-9, 11-24, 26-38, 40-46, and 48-50 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. Claims 1, 17, 49, and 50 have been amended to include the limitation "always using/determining a lookup table entry". The specification does not disclose that this is always done. It is silent as to when this is done, except to teach that it does occur. Also, claims 1, 17, 49, and 50 have been amended to include the limitation "regardless of whether said lowest level contains a free segment of a size equal to or larger than the requested size" or something similar. The specification is silent as to anything being done or not done regardless of this condition. It is possible that at least in some cases the action being performed was conditioned on this condition.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-9, 11, 17-18, 23, 26-29, 31, 38, 40-42, 44-46, and 48-50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Masmano et al. ("Dynamic storage allocation for real-time embedded systems") in view of Moreno ("A Wrapper for Look-Up Tables (LUT) Operations in C++").

5. In regards to claim 1, Masmano teaches a method of using a computer having a CPU linked to a digital data memory for processing requests for allocation of a memory block of a data memory, wherein segments of the data memory are associated with different levels according to segment size, the method comprising:

6. (a) said CPU receiving a request for the allocation of a memory block within said digital data memory including receiving a binary data set indicative of requested memory block size (see size of 460 in section 6, paragraph 2) wherein each bit of the binary data set is associated with one of said levels ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1);

7. (b) said CPU determining a most significant set bit of the binary data set ("The first level index ... can be computed as the position of the last bit set (bit set to one) of the size."), section 6, paragraph 2) , and determining a lowest of said levels containing a segment of a size equal to or larger than the requested memory block regardless of whether said lowest level contains a free segment of a size equal to or larger than the requested memory block (calculating the second level index, section 6);

8. (c) said CPU determining, in the level determined in step (b), the availability of a free segment of a size equal to or larger than the requested memory block ("If this list is not empty then the block at the head of the list is removed from the list (marked as busy) and returned to the user", section 6, paragraph 6); and

9. (d) said CPU allocating a free segment depending on the determination in step (c) ("If this list is not empty then the block at the head of the list is removed from the list (marked as busy) and returned to the user", section 6, paragraph 6).

10. Masmano fails to teach an entry of a lookup table associated with one of said levels;

11. determining a lookup table entry associated with the most significant set bit, and

12. determining from said lookup table entry a lowest of said levels.

13. Moreno teaches using a lookup table to index predetermined values ("Look-Up Tables are a technique commonly used to accelerate numeric processing in applications with demanding timing requirements.", "The basic idea is to pre-compute the result of complex operations that are or can be expressed as a function of an integer value. The pre-computed results are typically stored in an array, which is used at run-time instead of performing the whole, time-consuming operation.", Introduction, paragraph 1 and 2) in order to reduce the overhead at run-time (id). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno to implement the calculation of the second level index using a lookup table such that an entry of a lookup table is associated with one of said levels;

14. determining a lookup table entry associated with the most significant set bit, and

15. determining from said lookup table entry a lowest of said levels

16. in order to reduce the overhead at run-time (id).

17. In regards to claim 2, Masmano further teaches (e) repeating steps (c) and (d) for a next higher level if no free segment of a size equal to or larger than the requested memory block has been found in step (c); and

18. (f) repeating step (e) until a free segment has been allocated or there is no next level ("search the next (bigger than the requested size) non empty list in the TSLF structure", section 6, paragraph 7).

19. In regards to claim 3, Masmano further teaches that each level is associated with a different granule size to the power of two, and sizes of memory segments allocated to a level are related to the granule size of the respective level ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1).

20. In regards to claim 4, Masmano further teaches that the granule size associated with a level defines a size difference between memory segments allocated to that level ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1).

21. In regards to claim 5, Masmano further teaches that step (a) further comprises rounding the requested memory block to a lowest granule size before performing steps (b) to (d) (see floor function in the formula determining the first level index, section 6, paragraphs 1-2).

22. In regards to claim 6, Masmano further teaches that each level is associated with a table of pointers indicative of memory addresses of free memory segments of a size allocated to the respective level (see figure 1).

23. In regards to claim 7, Masmano further teaches returning a pointer to the allocated free segment ("This function returns a pointer to a free block of the required size or bigger.", section 6, paragraph 5).

24. In regards to claim 8, Masmano further teaches returning a null pointer if no free segment is allocated ("In non-real time systems applications can receive a null pointer or are just killed by the OS when the system runs out of memory", section 2, paragraph 5).

25. In regards to claim 9, Masmano further teaches a bitmap is indicative of a state of memory segments (free, allocated) ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1), the bitmap comprising a root bitmap (first-level array's bitmap, figure 1 and section 5, paragraph 1), each bit of the root bitmap being indicative of whether or not an associated one of said levels contains at least one free segment ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1), and wherein step (b) further comprises determining from the root bitmap said lowest level containing a segment of a size equal to or larger than the requested memory block ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1).

26. In regards to claim 11, Masmano further teaches that each mask of a set of predetermined masks is associated with one of said levels, and step (c) further comprises performing a logic operation on the mask associated with the lowest level determined in step (b) and said binary data set ("The second level index can be obtained using a mask on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2), wherein a result of said logic operation is an index to bits of the bitmap indicative of the state of a segment of a size equal to or larger than the requested memory block (second-level array bitmap, figure 1 and section 5, paragraph 1).

27. In regards to claim 17, Masmano teaches a method of using a computer having a CPU linked to a digital data memory for managing a data memory, the method comprising:

28. said CPU defining a number of levels of the digital data memory;

29. said CPU defining a different rang of memory segment sizes for each level ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1); and

30. wherein a bit of a binary data set indicates size of a requested memory block (see size of 460 in section 6, paragraph 2), whereby a request for allocation of a memory block is processable by determining a level containing segments of a size equal to or larger than the requested memory block ("Most TLSF internal operations relay [sic] on the segregate_list() mapping function which given the size of a block calculates the indexes of the two arrays that points to the corresponding segregate list.", section 6, paragraph 1), and allocating a segment of the same size or larger than the

requested memory block in that level regardless of whether the determined level contains a free segment of the same size or larger than the requested memory block ("If this list is not empty then the block at the head of the list is removed from the list (marked as busy) and returned to the user", section 6, paragraph 6).

31. Masmano fails to teach said CPU generating a lookup table, wherein each entry of the lookup table is associated with a bit of a binary data set, and wherein each entry of the lookup table indicates one of the levels, whereby a request for allocation of a memory block is processable by always using the lookup table. Moreno teaches said CPU generating a lookup table and using a lookup table to index predetermined values ("Look-Up Tables are a technique commonly used to accelerate numeric processing in applications with demanding timing requirements.", "The basic idea is to pre-compute the result of complex operations that are or can be expressed as a function of an integer value. The pre-computed results are typically stored in an array, which is used at run-time instead of performing the whole, time-consuming operation.", Introduction, paragraph 1 and 2) in order to reduce the overhead at run-time (id). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno to implement the calculation of the second level index using a lookup table such that said CPU generating a lookup table, wherein each entry of the lookup table is associated with a bit of a binary data set, and wherein each entry of the lookup table indicates one of the levels, whereby a request for allocation of a memory block is processable by always using the lookup table in order to reduce the overhead at run-time (id).

32. In regards to claim 18, Masmano further teaches that the granule size defines a size difference between memory segments in each level ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1).

33. In regards to claim 23, Masmano further teaches generating a table of pointers for each level indicative of memory addresses of free memory segments of a size associated with the respective level (see figure 1).

34. In regards to claim 26, Masmano further teaches generating a set of masks, wherein each of the set of masks is associated with one of said levels, and wherein a logical operation of a binary data set indicative of the size of the requested memory block and the mask associated with a level containing segments of the same size as or larger than the requested memory block results in an index to a segment of a size the same as or larger than the requested memory block in that level ("The second level index can be obtained by using a mask on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2).

35. In regards to claim 27, Masmano further teaches creating a first doubly linked list ("Therefore, each free block is linked in two double linked lists", section 5, paragraph 3) of consecutive memory segments irrespective of size and status (free, allocated) ("In order to coalesce easily free blocks, the TLSF employs the boundary tag technique proposed by D. Knuth in [3] which consists on [sic] adding a footer field to each free or used block which is a pointer that points to the start of the same block.", section 5, paragraph 3); and

36. creating a second doubly linked list of free memory segments of the same size (see list shown in figure 1).

37. In regards to claim 28, Masmano further teaches that memory segments in the first doubly linked list are arranged in order of associated memory addresses ("In order to coalesce easily free blocks, the TLSF employs the boundary tag technique proposed by D. Knuth in [3] which consists on [sic] adding a footer field to each free or used block which is a pointer that points to the start of the same block.", section 5, paragraph 3).

38. In regards to claim 29, Masmano further teaches when freeing a memory segment:

39. determining a state of memory segments adjacent to the memory segment to be freed using the first doubly linked list;

40. merging the memory segment to be freed with free adjacent memory segments; and

41. updating the first and second doubly linked lists accordingly ("When a block is free the footer of the previous block (which is located one word before the freed block) is used access the head of the block to check whether it is free or not and merge both blocks accordingly.", section 5, paragraph 3).

42. In regards to claim 31, Masmano further teaches updating the second doubly linked list upon allocation of a memory segment upon request ("Each array of lists has an associated bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1).

43. In regards to claim 38, Masmano further teaches freeing and allocating segments of the data memory: the method comprising, when freeing a memory segment:

44. determining a state of memory segments adjacent to the memory segment to be freed; and

45. merging the memory segment to be freed with free adjacent memory segments ("When a block is freed the footer of the previous block (which is located one word before the freed block) is used access the head of the block to check whether it is free or not and merge both block accordingly.", section 5, paragraph 2).

46. In regards to claim 40, Masmano further teaches an operating system for a computer ("OS", section 2, paragraph 5), adapted to perform the method of claim 1.

47. In regards to claim 41, Masmano further teaches that the operating system is a realtime operating system ("real-time systems", section 1, paragraph 2).

48. In regards to claim 42, Masmano further teaches that the method described above is performed at task level ("applications, section 2, paragraph 5).

49. In regards to claim 44, Masmano further teaches a computer program adapted to perform the method of claim 1 when operated on a computer ("hardware used in embedded systems", section 4, paragraph 3).

50. In regards to claim 45, Masmano further teaches a storage medium having stored thereon a set of instruction, which when executed by a computer, performs the method of claim 1 (test code shown in figure 2).

51. In regards to claim 46, Masmano further teaches a computer system comprising the operating system of claim 40 ("embedded systems", title).

52. In regards to claim 48, Masmano further teaches defining a different granule size for each level, wherein the size of each memory segment is related to the granule size of the respective level ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1).

53. In regards to claim 49, Masmano teaches a method of using a computer having a CPU linked to a digital data memory for processing requests for allocation of a memory block of a data memory, wherein segments of the data memory are associated with different levels according to segment size, the method comprising:

54. (a) said CPU receiving a request for the allocation of a memory block within said digital data memory including receiving a binary data set indicative of requested memory block size (size, section 6, paragraph 2) wherein each bit of the binary data set is associated with one of said levels ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1);

55. (b) said CPU determining the lowest of said levels containing a segment equal to or larger in size than the requested memory block including determining a most significant set bit of the binary data set (first level index, section 6, paragraph 2), and always determining based on the most significant set bit a lowest of said levels containing a segment of a size equal to or larger than the requested memory block regardless of whether said lowest level contains a free segment of a size equal to or larger than the requested memory block ("calculate the 'f' and 's' indexes (using the mapping function) which are used to get the head of the free list holding the closer class list", section 6, paragraph 6);

56. (c) providing a bitmap indicative of a state of memory segments (free, allocated) (bitmap associated with second-level, section 5, paragraph 1);

57. (d) providing a plurality of masks, wherein each mask is associated with one of said levels ("The second level index can be obtained using a mask on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2);

58. (e) said CPU using the most significant set bit to index a mask associated with said lowest of said levels containing a segment of a size equal to or larger than the requested memory block ("The second level index can be obtained using a mask on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2);

59. (f) said CPU logically combining said indexed mask with said binary data set to output an index to said bitmap ("The second level index can be obtained using a mask on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2); and

60. (g) said CPU determining, from said bitmap, the availability of a free segment of a size equal to or larger than the requested memory block using said index to said bitmap ("search the next (bigger than the requested size) non empty list in the TSLF structure. ... If a list is found, then the block at the head of the list will be used to fulfill the request.", section 6, paragraph 7).

61. Masmano fails to teach a lookup table with entries associated with each of said levels used to index a mask based on the most significant set bit; and that the plurality of masks are predetermined. Moreno teaches using a lookup table to index predetermined values ("Look-Up Tables are a technique commonly used to accelerate numeric processing in applications with demanding timing requirements.", "The basic

idea is to pre-compute the result of complex operations that are or can be expressed as a function of an integer value. The pre-computed results are typically stored in an array, which is used at run-time instead of performing the whole, time-consuming operation.", Introduction, paragraph 1 and 2) in order to reduce the overhead at run-time (id). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno to include a lookup table with entries associated with each of said levels used to index a mask based on the most significant set bit; and that the plurality of masks are predetermined in order to reduce the overhead at run-time (id).

62. In regards to claim 50, Masmano teaches a method of using a computer having a CPU linked to a digital data memory for managing a data memory, the method comprising:

63. said CPU defining a number of levels of the digital data memory ("In order to speedup the access to the free blocks and also to manage a large set of segregated lists (to reduce fragmentation) the array or lists has been organized as a two level array.", section 5, paragraph 1);

64. said CPU defining a different range of memory segment sizes for each level ("The first-level array divides free blocks in classes that are a power of two apart (8, 16, 32, 64, etc.)", section 5, paragraph 1); and

65. said CPU generating masks, wherein each mast is associated with a bit of a binary data set that indicates size of a requested memory block, and wherein each mask indicates one of the levels ("The second level index can be obtained using a mask

on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2), whereby a request for allocation of a memory block is processable by determining a level containing segments of a size equal to or larger than the requested memory block by always using masks, and allocating a segment of the same size or larger than the requested memory block in that level regardless of whether the determined level contains a free segment of the same size or larger than the requested memory block ("calculate the 'f' and 's' indexes (using the mapping function) which are used to get the head of the free list holding the closer class list", section 6, paragraph 6), said method including:

66. (a) said CPU receiving a request for the allocation of a memory block within said digital data memory including receiving a binary data set indicative of requested memory block size (size, section 6, paragraph 2) wherein each bit of the binary data set is associated with a mask associated with one of said levels ("The first-level array divides free blocks in classes that are a power of two apart (8, 16, 32, 64, etc.)", section 5, paragraph 1);

67. (b) said CPU determining the lowest of said levels containing a segment equal to or larger in size than the requested memory block including determining a most significant set bit of the binary data set ("The first level index ... can be computed as the position of the last bit set (bit set to one) of the size.", section 6, paragraph 2), and determining from the mask associated with the most significant set bit a lowest of said levels containing a segment of a size equal to or larger than the requested memory

block ("search the next (bigger than the requested size) non empty list in the TSLF structure", section 6, paragraph 7);

68. (c) said CPU determining, in the level determined in step (b), the availability of a free segment of a size equal to or larger than the requested memory block ("If a list is found, then the block at the head of the list will be used to fulfill the request.", section 6, paragraph 7);

69. (d) said CPU allocating a free segment depending on the determination in step (c) ("If a list is found, then the block at the head of the list will be used to fulfill the request.", section 6, paragraph 7); and

70. (e) said CPU determining, from said bitmap, the availability of a free segment of a size equal to or larger than the requested memory block using said index to said bitmap ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1).

71. Masmano fails to teach said CPU generating a lookup table wherein each entry of the lookup table is associated with a mask. Moreno teaches generating a lookup table wherein each entry of the lookup table is associated with a predetermined values ("Look-Up Tables are a technique commonly used to accelerate numeric processing in applications with demanding timing requirements.", "The basic idea is to pre-compute the result of complex operations that are or can be expressed as a function of an integer value. The pre-computed results are typically stored in an array, which is used at run-time instead of performing the whole, time-consuming operation.", Introduction, paragraph 1 and 2) in order to reduce the overhead at run-time (id). It would have been

obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno such that said CPU generates a lookup table wherein each entry of the lookup table is associated with a mask in order to reduce the overhead at run-time (id).

72. Claims 12-16 and 19-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Masmano et al. ("Dynamic storage allocation for real-time embedded systems") in view of Moreno ("A Wrapper for Look-Up Tables (LUT) Operations in C++") and Morzy et al. ("Hierarchical Bitmap Index: An Efficient and Scalable Indexing Technique for Set-Valued Attributes").

73. In regards to claim 12, Masmano further teaches that said bitmap comprises a plurality of second stage bitmaps (second-level array's bitmap, figure 1 and section 5, paragraph 1), each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps (see figure 1 and section 5, paragraph 1), and each bit of the second stage bitmap being indicative of whether or not an associated segment is free ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1), and wherein the operation result is an index to one bit of the second stage bitmap and said one bit of the second stage bitmap being indicative of the state of a segment of a size the same as or larger than the requested memory block ("The second-level sub-divides each first-level class linearly", "The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1). Masmano in view of Moreno fails to

teach a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps. Morzy teaches a plurality of third stage bitmaps (index key leaves, figure 1) and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps ("At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to '1'.", page 243, paragraph 1) in order to scalably and efficiently index large collections (page 250, paragraph 4). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine Masmano with Moreno and Morzy for a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps in order to scalably and efficiently index large collections (Morzy, page 250, paragraph 4).

74. In regards to claim 13, Masmano further teaches that if no free segment is found, repeating the determination for the next more significant bit of said predetermined number of bits of the third stage bitmap, until a free segment is found or there is no more significant bit of said predetermined number of bits of the third stage bitmap ("search the next (bigger than the requested size) non empty list in the TSLF structure", section 6, paragraph 7).

75. In regards to claim 14, Masmano further teaches that if no free segment is found, repeating the determination for the next more significant bit of said predetermined number of bits of the third stage bitmap, until a free segment is found or there is no

more significant bit of said predetermined number of bits of the third stage bitmap ("search the next (bigger than the requested size) non empty list in the TSLF structure", "If no bigger block is found, then", section 6, paragraph 7).

76. In regards to claim 15, McMahon further teaches that if no free segment is found, repeating the determination for the second stage bitmap associated with the next more significant set bit of the root bitmap, until a free segment is found or there is no more significant bit of the root bitmap ("search the next (bigger than the requested size) non empty list in the TSLF structure", "If no bigger block is found, then", section 6, paragraph 7).

77. In regards to claim 16, McMahon further teaches that each bit of the third stage bitmaps is associated with an entry in a table of pointers indicative of memory addresses of free memory segments (See figure 1).

78. In regards to claim 19, Masmano further teaches that the bitmap comprises a root bitmap (bitmap associated with first-level array, figure 1 and section 5, paragraph 1), each level being associated with one bit of the root-bitmap (see dashed lines, figure 1), and a plurality of second stage bitmaps associated with the segments (see dashed lines, figure 1), each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks", section 5, paragraph 1). Masmano in view of Moreno fails to teach a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps. Morzy

teaches a plurality of third stage bitmaps (index key leaves, figure 1) and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps ("At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to '1'.", page 243, paragraph 1) in order to scalably and efficiently index large collections (page 250, paragraph 4). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine Masmano with Moreno and Morzy for a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps in order to scalably and efficiently index large collections (Morzy, page 250, paragraph 4).

79. In regards to claim 20, Morzy further teaches that the bitmap comprises a root bitmap (index key root level, figure 1), each level being associated with one bit of the root-bitmap (arrows from index key root level, figure 1), and a plurality of second (inner nodes, figure 1) and third stage bitmaps (index key leaves, figure 1) associated with the segments, each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps ("In the root of the index key only first and third bits are set to '1', which means that only first and third inner nodes at the level 2 are non-empty.", page 243, paragraph 1), and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps ("At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to '1'.", page 243, paragraph 1).

80. In regards to claim 21, Masmano further teaches updating the bitmap when a segment is allocated ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1).

81. In regards to claim 22, Masmano further teaches updating the bitmap when a segment is freed ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1).

82. Claims 24 and 35-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Masmano et al. ("Dynamic storage allocation for real-time embedded systems") in view of Moreno ("A Wrapper for Look-Up Tables (LUT) Operations in C++") and McMahon et al. (US 5,784,699).

83. In regards to claim 24, Masmano further teaches generating a bitmap indicative of whether or not a level contains at least one free segment ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1),

84. wherein each bit of the third stage bitmaps is associated with an entry in the table of pointers (see figure 1).

85. Masmano fails to teach generating a bitmap indicative of a state of each segment (free, allocated). McMahon teaches generating a bitmap indicative of a state of each segment (free, allocated) (group bit map indexes, 350 and 360, figure 3C) in order to permit searching in an extremely small amount of memory (Col. 4, lines 42-44). It would have been obvious at the time the invention was made to a person of ordinary skill in

the art to combine Masmano with Moreno and McMahon by generating a bitmap indicative of a state of each segment (free, allocated) in order to permit searching in an extremely small amount of memory (id).

86. In regards to claim 35, Masmano further teaches allocating free segments of the data memory to different levels according to size ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1); and

87. providing a bitmap comprising different stages, wherein the bits of one stage are indicative of availability of free segments in said levels ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1).

88. Masmano in view of Moreno fails to teach that the bits of another stage are indicative of state and/or size and/or location of individual segments. McMahon teaches that the bits of another stage are indicative of state and/or size and/or location of individual segments (group bit map indexes, 350 and 360, figure 3C) in order to permit searching in an extremely small amount of memory (Col. 4, lines 42-44). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno and McMahon such that the bits of another stage are indicative of state and/or size and/or location of individual segments in order to permit searching in an extremely small amount of memory (id).

89. In regards to claim 36, Masmano further teaches that the bits of one stage are associated with pointers indicative of a memory address of free segments (see figure 1).

90. In regards to claim 37, Masmano further teaches updating the bitmap to reflect the allocation or release of memory segments ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1).

91. Claims 30 and 32-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Masmano et al. ("Dynamic storage allocation for real-time embedded systems") in view of Moreno ("A Wrapper for Look-Up Tables (LUT) Operations in C++") and Wilson et al. ("Dynamic Storage Allocation: A Survey and Critical Review").

92. In regards to claim 30, Masmano in view of Moreno teaches claim 27. Masmano in view of Moreno fails to teach that each second doubly linked list is a LIFO (Last In First Out) list. Wilson teaches that each second doubly linked list is a LIFO (Last In First Out) list ("Recently-freed blocks would therefore be "first," and tend to be reused quickly, in LIFO (last-in-first-out) order.", page 44, paragraph 4) because "freeing is very fast" (page 44, paragraph 4). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno and Wilson such that each second doubly linked list is a LIFO (Last In First Out) list because "freeing is very fast" (id).

93. In regards to claim 32, Masmano in view of Moreno teaches claim 27. Masmano in view of Moreno fails to teach if a segment determined for allocation upon request is larger than a requested memory block:

94. allocating a portion of the determined segment large enough to satisfy the request;

95. providing a remaining portion as a new free memory segment; and

96. updating the first and second doubly linked lists accordingly.

97. Wilson teaches if a segment determined for allocation upon request is larger than a requested memory block ("the free list for the corresponding size class is searched for a block at least large enough to hold it.", page 53, paragraph 3):

98. allocating a portion of the determined segment large enough to satisfy the request ("the free list for the corresponding size class is searched for a block at least large enough to hold it.", page 53, paragraph 3);

99. providing a remaining portion as a new free memory segment ("An allocated block", page 1, paragraph 4); and

100. updating the first and second doubly linked lists accordingly ("Each block of memory has a both header and a 'footer' field, both of which record the size of the block and whether it is in use.", page 39, paragraph 7, "One of the simplest allocators uses an array of free lists, where each list holds free blocks of a particular size", page 51, paragraph 6).

101. It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno and Wilson such that if a segment determined for allocation upon request is larger than a requested memory block:

102. allocating a portion of the determined segment large enough to satisfy the request;

103. providing a remaining portion as a new free memory segment; and

104. updating the first and second doubly linked lists accordingly

105. so that the unused portion can be subsequently used.

106. In regards to claim 33, Masmano further teaches that each segment is associated with a header, thereby to form the first doubly linked list, each header including information indicative of a state (free, allocated) of the associated segment and a pointer indicative of a memory address of the previous segment (see figure 1). Masmano in view of Moreno fails to teach that each header includes information indicative of the size of the associated segment. Wilson teaches that each header includes information indicative of the size of the associated segment ("Each block of memory has a both header and a "footer" field, both of which record the size of the block and whether it is in use.", page 39, paragraph 7). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno and Wilson such that each header includes information indicative of the size of the associated segment in order to easily find the next block.

107. In regards to claim 34, Wilson further teaches that the header associated with each free segment of a given size further includes a pointer indicative of the memory address of a previous and/or subsequent free segment of the same size, depending on the availability of a previous and/or subsequent free segment of the same size and in accordance with the order of free segments of the same size in the LIFO list ("For

allocators using free lists or indexing trees to keep track of free blocks, the list or tree nodes are generally embedded in the free blocks themselves.", page 40, paragraph 3).

108. Claim 43 is rejected under 35 U.S.C. 103(a) as being unpatentable over Masmano et al. ("Dynamic storage allocation for real-time embedded systems") in view of Moreno ("A Wrapper for Look-Up Tables (LUT) Operations in C++") and Rubini et al. (*Linux Device Drivers*).

109. In regards to claim 43, Masmano in view of Moreno teaches the operating system of claim 40. Masmano in view of Moreno fails to teach performing the method described above at interrupt level. Rubini teaches performing memory allocation at interrupt level ("GFP_ATOMIC Used to allocate memory from interrupt handlers and other code outside of a process context. Never sleeps.", section 7.1.1) because it never sleeps (section 7.1.1). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno and Rubini by performing the method described above at interrupt level because it never sleeps (section 7.1.1).

Response to Arguments

110. Applicant's arguments with respect to claims 1-48 have been considered but are moot in view of the new ground(s) of rejection.

111. Applicant's arguments with respect to claims 49 and 50 have been fully considered but they are not persuasive. These arguments are directed toward

overcoming a rejection based on McMahon while the rejection under 103 of claims 49 and 50 in the office action mailed September 16, 2010 was based on Masmano and Moreno. These references are sufficiently different that the arguments relating to McMahon do not apply to Masmano and Moreno.

Conclusion

112. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

113. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

114. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Nathan Sadler whose telephone number is (571)270-7699. The examiner can normally be reached on Monday - Thursday 8:30-6:00 EST.

115. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Reginald Bragdon can be reached on (571)272-4204. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

116. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/N. S./
Nathan Sadler
Examiner, Art Unit 2189
April 4, 2010

/Reginald G. Bragdon/
Supervisory Patent Examiner, Art Unit 2189